07

Filtrando usando mayor que menor que y diferente

Transcripción

[00:00] Bien, ya aprendimos entonces a realizar filtros simples únicamente seleccionando ciertos campos con un valor definido, pero podemos también crear filtros utilizando la condición mayor que, menor que, igual o diferente de. Entonces creemos otro script.

[00:21] Entonces vamos SELECT * FROM, asterisco siempre es al. SELECT * FROM tbcliente. Volvamos a la tabla de cliente, entonces vamos a correr. Entonces veamos aquí, tenemos también el campo de edad, entonces vamos a copiar SELECT * FROM tbcliente; WHERE EDAD = No. EDAD mayor que, vamos a ver qué edades hay aquí. 26. 27. Edad mayor a 27.

[01:17] Entonces lo seleccionamos y presionamos el rayo y nos devuelve los clientes con la edad mayor a 27 años. Ahora si yo quiero, bueno, las edades menores. Entonces sería aquí menor que 27. Entonces corremos este comando y me devuelve a todos los que son menores de 27. Si yo le digo menor o igual aquí a 27, sería el orden igual o menor a 27. Menor o igual va en este orden.

[01:55] Entonces, yo voy a también traer el 27, si existiera. Como no hay nadie de 27 años, pues no me lo trae. Ahora, si yo quiero diferente, la edad diferente a 27, la sintaxis para diferente en MySQL es así. Entonces menor que, mayor que. Le damos y me devuelve a todas las personas, diferente de 27 años, pero como no tengo de 27, pues lógicamente me lo va a devolver a todos, pero si pongamos 29, aquí hay una persona que tiene 29 años, o 26, que vi los registros aquí, más sencillo, 26.

[02:40] Entonces seleccionamos y le damos al rayo aquí nuevamente y nos devuelve todas las edades, excepto los que tienen 26 años. Eso por un lado. Ahora bien, ¿será que MySQL logra identificar mayor que y menor que con nombres por ejemplo? Veamos. SELECT * FROM tbcliente WHERE.

[03:18] Recuerden aquí que mayúscula o minúscula no hace ninguna diferencia, pero la buena práctica nos dice que el comando es mejor siempre mantenerlo en mayúscula. Entonces, WHERE NOMBRE > digamos aquí este señor, Abel Pintos. No. Erica Carbajo. Listo, entonces mayor que Erica. 'Erica Carbajo'; bien. Entonces, punto y coma.

[03:54] ¿Será que sí nos devuelve algo? Veamos. Entonces sí, MySQL tiene una forma también de organizar este tema alfabéticamente. Entonces mayor que Erica, entonces serían todos los nombres que estén después de la E. Si yo por ejemplo tengo Erica y tengo Enrique, la N viene antes de la R, entonces Enrique no me va a aparecer.

[04:22] Veamos aquí abajo para que ustedes observen. A ver si hay algún Enrique menor que Erica. Les damos al rayo y entonces nos devuelve Abel, Carlos, Edson, Edson, la D viene antes de la R, entonces obviamente, pero yo puedo darle también menor o igual a Erica Carbajo.

[04:48] Lógicamente me va a devolver a Erica también. Ahora volvamos aquí a nuestra tabla de producto. SELECT * FROM tbproducto; y les voy a mostrar aquí algo que puede suceder, que puede generar un poco de confusión y es por ejemplo, aquí precio lista.

[05:14] Digamos, precio lista está definido como un float, no establecimos un número exacto, un decimal, por ejemplo, con dos casillas decimales y por ejemplo yo aquí en vez de tener 4.9, lo lógico sería 4.90 centavos, pero no hay ningún problema. Entonces yo quiero por ejemplo, veamos a ver si aquí tenemos un float con tres números. Supongamos este de 28.51, entonces digamos aquí: SELECT * FROM tbproducto WHERE PRECIO_LISTA = 28.51;

[06:12] Vean lo que va a suceder: no nos va a devolver nada. Estamos hablando de coma flotante, digamos aquí esta situación nos impide que él identifique ese precio, a pesar de que nosotros lo visualizamos ahí. No lo logra identificar exactamente así. Yo puedo tratar de digamos establecer. Puedo sí ver mayores que éste, lógicamente, los voy a ver, mayores que 28.51.

[06:48] Pero me lo va a devolver, y observen ustedes qué curioso, ahí sí me devuelve el 28.51, es coma flotante, entonces aparece, eso es un poco extraño que suceda, pero es un asunto de MySQL. No hay una forma de pronto de observarlo vean, por ejemplo aquí, si yo le digo menores de 28.51, observemos, ahí ya no nos lo devuelve. ¿Por qué?

[07:18] Porque tal vez ese 28.51 tenga unos números adicionales allí que esté considerando nuestro algoritmo para determinar si es mayor o si es menor. Entonces, en este caso digamos, si queremos hallar exactamente este valor, lo que tenemos que hacer es aplicar una condición adicional. Y esa condición adicional es con un comando llamado between, establecemos un rango de valores.

[07:51] Entonces SELECT * FROM tbproducto WHERE PRECIO_LISTA BETWEEN, entonces vamos a colocar un límite inferior y un límite superior, BETWEEN, entonces 28.48, bien cercanos, AND 28.52; punto y coma y rodamos, y nos va a devolver exactamente el 28.51.

[08:24] Bien. Básicamente esta es una forma de traer exactamente un registro cuando se trata de un float. Para evitar este problema, lo interesante sería definir, por ejemplo, el precio como un número decimal y establecer con dos casillas decimales y el número posible del precio como tal. Ahí ya sería un asunto más de configurar bien ese parámetro de modo que no se presente esta situación.

[08:53] En el próximo video estaremos entonces viendo un poco más a fondo el tema de filtros compuestos.